Openmixer: a routing mixer for multichannel studios

Fernando Lopez-Lezcano, Jason Sadural
CCRMA, Stanford University
Stanford, CA, USA
nando@ccrma.stanford.edu, jsadural@ccrma.stanford.edu

Abstract

The Listening Room at CCRMA, Stanford University is a 3D studio with 16 speakers (4 hang from the ceiling, 8 surround the listening area at ear level and 4 more are below an acoustically transparent grid floor). We found that a standard commercial digital mixer was not the best interface for using the studio. Digital mixers are complex, have an opaque interface and they are usually geared towards mix-down to stereo instead of efficiently routing many input and output channels. We have replaced the mixer with a dedicated computer running Openmixer, an open source custom program designed to mix and route many input channels into the multichannel speaker array available in the Listening Room. This paper will describe Openmixer, its motivations, current status and future planned development.

Keywords

Surround sound, studio mixer, spatialization, ambisonics.

1 Introduction

The Listening Room was created when The Knoll (the building that houses CCRMA) was completely remodelled in 2005. It is a very quiet studio with dry acoustics designed for research, composition and diffusion of sounds and music in a full 3D environment. Digital mixers proved to be less than optimal for using the room and the Openmixer project was started to write an application from scratch that would be a better match for our needs.

The need for an alternative solution to digital mixers was recognized by Fernando Lopez-Lezcano in 2006/2007 and he did some research into existing software based alternatives together with Carr Wilkerson at CCRMA. None of the options that were found quite met the goals we had in mind. It was only last year (2009) that the project really started taking off thanks to the enthusiasm and dedication of Jason Sadural (a student at CCRMA) and the support of Jonathan Berger and Chris Chafe. Both Jason and Fernando have been working on hardware and software design and coding since then.

The Listening Room

2 Digital Mixers and the Listening Room

Existing digital mixers are very versatile but are generally not a good fit for a situation that needs many input and output channels. Most of them are designed for mix-down to stereo or at most for mix-down to a fixed 5.1 or 7.1 surround environment. If more output channels are needed expansion slots have to be used and the physical size and price of the mixer goes up very fast. It is also difficult to find mixers with the capabilities we needed and without fans.

Another usability problem of digital mixers is preset management. While it is possible to save and load the mixer state to presets (a feature absolutely necessary in our multi-user shared environment), digital mixers don't have a user database or per-user password protection, at least in mixers that are in our price range. Presets can be

protected but there is nothing to prevent someone from changing other user's presets by mistake.

Digital mixers have opaque interfaces, with options sometimes buried deeply inside a layered menu structure. It is sometimes very hard to know why a particular patch is not working.

We originally installed a Tascam DM3200 in the Listening Room, using two expansion cards to barely provide enough input/output channels (the 3200 has 16 buses and we would like to have a maximum of 24 mixer outputs). In addition to the generic digital mixer problems we found out that switching presets in the 3200 would cause clicks in the analog outputs of the expansion boards that were driving the speakers. How this can happen in a professional mixer is beyond our understanding.

3 Requirements

What would be the ideal sound diffusion system for the Listening Room?

The user interface should be very simple. Nothing should be hidden inside hierarchical menus. All controls should directly affect the diffusion of sound. It should be easy to learn to use it (intuitive). It should be possible to load and save presets using the same authentication system that is used to login in all computers at CCRMA. It should also be possible to control all aspects of the mixer remotely over the network using OSC (Open Sound Control).

The system should support a routing and level control matrix with many multichannel input streams (up to 24 channels wide) coming from different types of sources (analog, digital, network) and many analog outputs (up to a maximum of 24 channels). The system should be able to also deal with multichannel input streams already encoded in Ambisonics, and give the user a choice of predefined decoders calibrated to the speaker array installed in the room.

The system does not need to have audio processing tools such as limiters, compressors, equalization, effects, etc. It should be as transparent as possible. Anything that complicates the user interface should be trimmed down.

It has to be a stand-alone system, always running, and users should not need to login into a computer to start a custom mixer application (for example they should be able to play multichannel content directly from the dvd or blue ray player).

4 A solution

One solution is to use a general purpose computer and off the shelf components as the heart of the mixer, and write a program that orchestrates the process of routing, mixing and decoding the audio streams. A Linux based workstation running Jack [1] and Planet CCRMA is the high performance, low latency and open platform in which we based the design.

Other projects have implemented software mixing and sound diffusion in a general purpose computer, examples include the very capable SuperCollider based software mixer written by Andre Bartetzki [2] at TU Berlin. Or the ICAST [3] sound diffusion system at the Louisiana State University. Or the BEAST system [4] at Birmingham.

Our needs are much more modest and require a system that is based on a single computer that can directly boot into a fully working mixer system.

4.1 The hardware

The Listening Room is a very quiet room with dry acoustics. It is essential it remains quiet so one of our fanless workstations [5] is used as the Openmixer computer. It is a high performance quad core computer with 8Gbytes of RAM and should be able to cope with all reasonable future needs for mixing and processing (see Figure 1 for an overview of the hardware).

4.1.1 Input / output requirements

This is what we need to have available: 16 channels of analog line level balanced input, 8 channels of microphone level input, 8 channels of line level audio input for a media player (DVD/Blu Ray), 3 ADAT lightpipes for direct connection of the audio workstation in the Listening Room (another custom made fanless computer), 3 ADAT lightpipes available for external computers, and dedicated Gigabit Ethernet network jacks with a DHCP server and netjack or jacktrip software for multichannel network audio sources (up to 24 channels per source). All external connections (analog, ADAT, network) should be available in a rack mounted patch panel.

4.1.2 Audio input / output hardware

The core of the audio interface is an RME MADI PCI express soundcard connected through MADI to a Solid State Logic XLogic Alpha-Link MADI

AX box. This combination alone provides for 64 i/o channels from the Openmixer computer including 24 high quality analog inputs and outputs and 3 ADAT lightpipes.

Additional ADAT lightpipes for external computers are provided by an RME RayDAT sound card. This card is currently not installed as we are waiting for the newer implementation of the RME driver for Linux - the existing driver does not support the word clock daughter card that we need to use to synchronize the RayDAT with the main RME MADI card using Word Clock.

The 24 analog outputs are reserved for direct speaker feeds. 16 of the 24 analog inputs are connected to a front panel analog patchbay and the other 8 are connected to the media player. Currently 2 of the 3 ADAT lightpipes go to the audio workstation and 1 ADAT lightpipe is fed from an 8 channel mic preamp with connections to the analog patchbay (this setup will change when the RayDAT card is again part of the system).

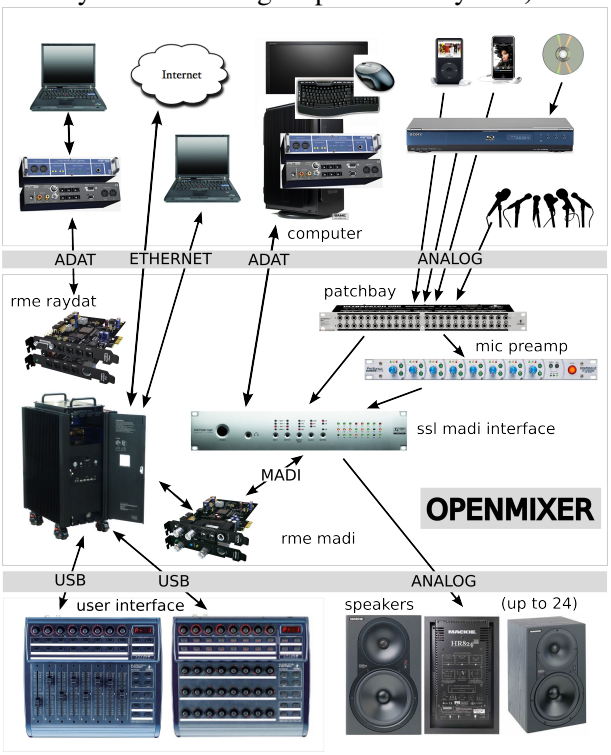

Figure 1: Hardware Overview

The computer has two ethernet interfaces, one connects to the Internet and the other provides DHCP enabled dedicated Gigabit switch ports for laptops or other wired computer systems that connect to Openmixer through the network using either Jacktrip or Netjack.

The RME MADI card is the master clock source for the system, everything else slaves to it through Word Clock or ADAT Sync.

4.1.3 User interface hardware

The only (for now) interface to Openmixer is through a couple of USB connected dedicated MIDI controllers. The current controllers are a Behringer BCF2000 for the main controller and a BCR2000 for the secondary routing controller. Those were selected based on having enough functionality for our needs and being very cheap (and thus easy to replace if they break down). Other more robust controllers might be used in the future. As Openmixer is just software it is easy to switch to a slightly different controller with simple modifications to the source code.

4.2 The software

4.2.1 Choosing a language and environment

Initial proposals included a simple implementation using Pd but we did not go that way because a system based on a text language seemed easier to be debugged and could be easily booted without needing monitors, keyboard or a mouse to interact with it. Another option that was explored was to program Openmixer in C and/or C++ and borrow code from other compatible open source projects or libraries (Openmixer is released under the GPL license). This would have given us complete control at a very low level and would probably be the most efficient implementation, but it would have been more complex to code compared with our final choice. This approach was discarded and simplicity won over efficiency. Another option briefly discussed was to actually use Ardour as the engine for the mixer (after all that is what Ardour does best!), but adapting such a complex project to our needs would have also been very time consuming. In the end we settled for the SuperCollider [6] language as it is text based and can handle most of the needs of the project by itself without the need to use any other software. It can handle audio processing and routing very easily and efficiently, it can send and receive MIDI and OSC messages, and even use the SwingOSC SuperCollider classes for a future GUI display. Of course Jack is used as the audio engine, as well as several support programs that are started and controlled from within SuperCollider (amixer,

jack_connect, jack_lsp, ambdec_cli, jconvolver, etc).

4.3 The user interface

The current interface is layered on top of the capabilities of the BCF2000/BCR2000 controllers. See Figure 2 for an overview of the main controls.

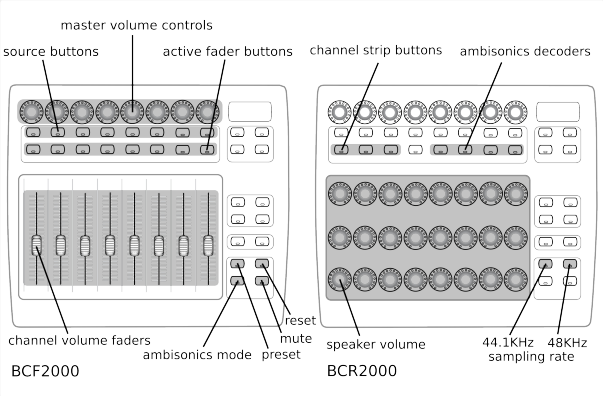

Figure 2: User interface

There are currently two modes of operation already implemented and being used:

4.3.1 Normal Mode

Normal Mode (see Figure 3) routes and controls volume levels of any channel of any input source to a single speaker or a set of speakers. Each input channel from a given input source is associated with a fader in the BCF2000, and can be routed to a single speaker or a set of speakers through the speaker volume knobs in the BCR2000. The levels of a particular input channel can be controlled through the fader {2}, the volume through each speaker can be controlled though the corresponding speaker knob, {3} and a master volume {5} can control all channels from a given source at the same time.

Diffusion of sources is simple to learn and can be done as follows:

1. Press the *Source Button* {1} for the desired input (audio workstation, analog inputs, microphone inputs, media player, network sources, etc).
2. Press one of the *Active Fader Buttons* {2} to activate it, or move the selected fader {2}. The *Active Fader Button* will become lit and the fader can be used to set the overall volume for that channel.

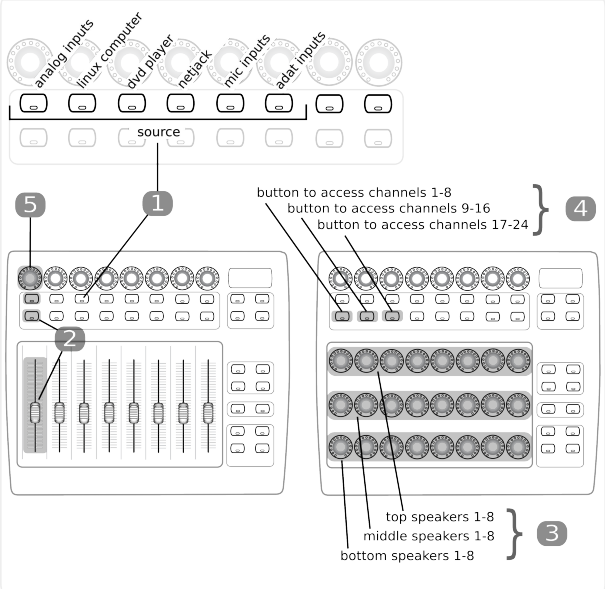

Figure 3: Normal mode

3. Turn up individual *Speaker Volume Knobs* {3} to adjust speakers levels for that channel.
4. Repeat steps 2 and 3 for each input channel. For input sources that have more than 8 channels use the *Channel Strip* buttons {4} in the BCR2000 to access additional input channel faders up to a maximum of 24 input channels for each source (a wider control surface would not need this extra step).
5. Control overall volume for that source with the *Master Volume Control* {5} for that source.

Two keys allow you to set all levels to zero (*Reset*) or load a preset for the selected source (*Preset*) in which individual input channels are pre-routed to individual output speakers.

Two more keys also allow you to select the sampling rate of the whole system (currently only *44.1 KHz* and *48 KHz* are supported).

It should be noted that input sources are not mutually exclusive, they route audio all the time to the speakers. To use more than one at the same time just select several sources separately and adjust the levels appropriately. All faders and controls will reflect the current state of a source after the corresponding source button is pressed.

4.3.2 Ambisonics Decoder Mode

In “Ambisonics Decoder Mode” (see figure 4) the assumption is made that the multi-channel

input source is an Ambisonics encoded audio stream (arrangement of input channels and scaling coefficients are fixed for each source). By pressing the *Ambisonics Mode* button while in a "normal mode" source, all channels of audio from that input source will be directly routed to an Ambisonics decoder with the same level, and from there to the speaker array.

All faders for that source are disabled and set to zero except for fader #1 which is the master volume control for all channels of the encoded input source (if any of the faders besides fader #1 are adjusted, it will automatically be adjusted back to zero after a few seconds).

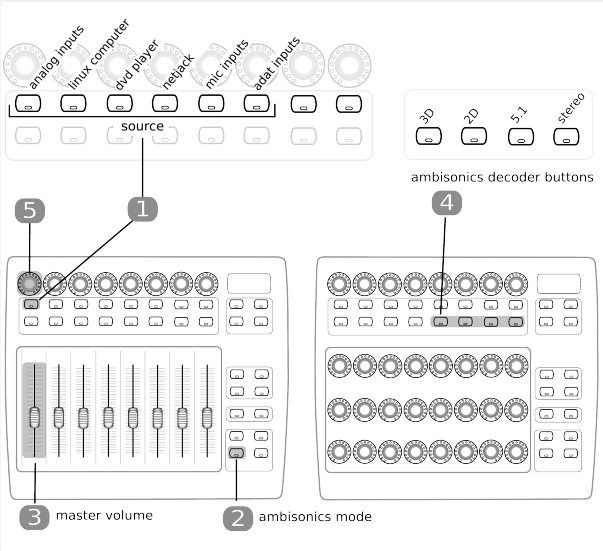

Figure 4: Ambisonics mode

Routing audio to the ambisonics decoders can be done as follows:

1. Select an input source {1} as you would in normal mode.
2. Press the *Ambisonics* decoder button {2} to route that input source directly to the Ambisonics decoders.
3. Adjust fader #1 {3} as the master volume control for the multichannel input source. All other fader or volume knobs are deactivated.
4. Choose the decoder {4} you would like to use.
5. Master mode volume controls {5} are set to unity by default and have exactly the same functionality as in Normal Mode.

4.3.3 Ambisonics Decoders

Openmixer uses *ambdec_cli* [7] and *jconvolver* [8] as external Jack programs that perform the Ambisonics decoding. Several decoders tuned to the speaker array are currently available:

1. 3D 2nd order horizontal, 1st order vertical
2. 2D third order horizontal
3. 2D 5.1 optimized decoder
4. Stereo (UJH decoding done through jconvolver)

This enables the composer or researcher to compare the rendering of the same Ambisonics encoded stream when it is decoded through several different decoders with varying order and capabilites, all properly tuned to the room and speaker arrangement.

When more speakers are added in the future the vertical order of the 3D decoder will be increased.

4.4 Network sources

An extremely simple and efficient way to connect external multichannel sound sources to the mixer is through a network connection. All laptop and desktop computers have a wired interface for high speed network connectivity and it would be perfect to use that for supplying audio to the mixer through the network (only one cable to connect).

There are currently two options:

4.4.1 Jacktrip

The Soundwire [9] project at CCRMA has created a software package (Jacktrip [10]) that is used for multi-machine network performance over the Internet. It supports any number of channels of bidirectional, high quality, uncompressed audio signal streaming. It would be ideal for our purposes except that it currently does not do sample rate synchronization between the computers that are part of a networked performance, so that if there is any clock drift between them there will be periodic clicks when the computers drift more than one buffer apart.

This is not of much concern in a remote network performance with high quality sound cards in both ends, as the quality of the network will usually create more significant problems, but it is crucial in a studio environment where all sources have to be in sync at the sample level (i.e.: no clicks allowed).

Still, Jacktrip is going to be supported with the idea of providing a remote (i.e.: not within the Listening Room) connection for jam sessions and small telepresence concerts.

4.4.2 Netjack

Netjack [11] is an ideal solution for a local network connection as the client computer does not use its own sound card at all (but it is possible to do that locally in the client computer with resampling being used to synchronize the two clocks), but just sends samples to an external Jack sink in the Openmixer computer. It is expected it will quickly become the preferred way to connect laptops and other external computers to the Openmixer system (as long as they can run Jack and Netjack).

The implementation of the Netjack connection in Openmixer was somewhat delayed due to the changing landscape of Netjack. The Openmixer computer is currently running Jack2 which had its own separate Netjack code base - one that relied on automatic discovery of the netjack server through multicast (which we don't currently have enabled at CCRMA). But there is now a backport of the Jack1 implementation of netjack to jack2 (*netone*) that would appear to be perfect for our needs.

The second ethernet interface of the Openmixer computer is connected to a small gigabit switch that provides local network jacks. The computer is set up to provide four DHCP ip addresses on that ethernet port and the firewall is structured to also route traffic to the Internet through the primary ethernet interface (NAT).

Four *jack_netsource* processes are spawned by SuperCollider, each one waiting for its DHCP ip address to become active with a netone jack client. When that happens the audio connection becomes active and Openmixer can control the volume and routing of all channels for that netjack source. For simplicity all potential network connected sources are currently mixed in parallel with equal gains (otherwise the user interface side of Openmixer would start to get more complicated).

5 Structure of the software

The software is written almost exclusively in SuperCollider. A boot time startup script written in perl is triggered when the computer enters into unix run level 5.

The script starts the SuperCollider sclang language interpreter, which takes care of setting up and monitoring the rest of the system. The perl script waits for the interpreter to finish (it should never exit in normal conditions) and restarts it if necessary. This can deal with unexpected problems that could cause SuperCollider's sclang to stop prematurely and it also implements a “system restart” function useful for development and debugging the system (a user can “touch” a file that triggers an orderly shutdown of the SuperCollider portion of Openmixer, and removing that file makes the perl script start it again). The perl script also logs the output of the SuperCollider program for debugging.

Once the SuperCollider sclang interpreter starts, it takes control of the rest of the Openmixer startup process. A number of external programs are used to control the hardware. The mixer of the RME card is initialized using *amixer*, and *jackd* is started with the appropriate parameters (currently jack runs with 2 periods of 128 frames – aproximately 5.3 mSecs of latency - but with the current cpu and load it should be possible to run at 64 frames if the lower latency is deemed necessary). After Jack is up and running two instances of the SuperCollider synthesis server *scsynth* are started (to spread the audio processing load between different cores of the processor), the Ambisonics decoders (*ambdec_cli* and *jconvolver*) are started and finally the *jack_netsource* processes for netjack inputs are spawned. Everything is connected together in the Jack world using calls to *jack_connect*. Finally the SuperCollider MIDI engine is initialized.

Once audio and midi are initialized the SuperCollider Synths that comprise the audio processing section of Openmixer are started and finally the SuperCollider MIDI and OSC responders that tie user controls to actions are defined and started. At this point Openmixer is operational and the user can start interacting with it.

Changing sampling rates is done by shutting down all audio processing and Jack, and restarting everything at the new sampling rate.

Periodic processes check for the presence of the USB MIDI controllers and initializes and reconnects them if necessary (that makes it possible to disconnect them and reconnect them to Openmixer without affecting the system).

Openmixer is currently around 2000 lines of SuperCollider code.

Much needs to be done, some parts of the code need work and reorganizing, and code needs to be added to make the system more reliable. All external programs will be monitored periodically by SuperCollider processes and restarted and reconnected if necessary. Depending on which program malfunctions the audio processing may be interrupted momentarily, but at least the system will recover automatically from malfunctions. The worst offender would be Jack, if it dies the whole system would need to be restarted (as it also happens when, for example, the sampling rate of the system is changed). If SuperCollider itself dies the perl script that starts up the system will restart it. The system is expected to be very stable as the computer in which Openmixer is running is dedicated to that purpose alone (ie: users are not logged in and running random software in it).

6 Future Development

6.1 Encoding to Ambisonics and VBAP

This is a different mode of operation in which individual channels of a given source can be positioned in space through Openmixer instead of being spatialized in the source through a separate program. In this case the second controller (BCR2000) is used to set elevation and azimuth angles in 3D space for each input channel instead of controlling the volume of individual speakers.

This has already been implemented for the case of Ambisonics but we have not yet made it available to end users.

An alternative implementation would do the same thing but using VBAP for spatialization instead of Ambisonics.

All parameters of the spatialization can also be controlled though OSC from a computer connected to the network.

6.2 Start-up sequence

The start-up sequence of the SuperCollider code needs to be changed so that the MIDI subsystem is initialized at the very beginning. That will enable us to use the control surfaces to give feedback to the user while Openmixer starts. Currently there is no way to know how far in the start-up sequence the software is, and when it is available for use (except for the movement of the faders at the very end if their position is different from the default).

7 Future

Saving and recalling presets is a very important feature and is difficult to implement properly. As a stopgap measure Jason has written a very simple OSC responder that enables a user to send OSC messages that save and restore named presets from a common directory. For now the communication is unidirectional so it is not possible to list existing presets, and they can't be protected or locked, but it enables users to save and restore a complex set-up which would otherwise need to be redone from scratch each time a user starts using Openmixer.

Without a display and a keyboard it is difficult to design an interface that is easy to use (because ideally preset management should be tied to the LDAP user database for authentication). One possible option to explore would be to use a web server interface in the Openmixer computer so that preset management can be accessed through a web browser in any computer connected to the network in the Listening Room. A user would need to login with the CCRMA user name and password to be able to access the web site. After that presets would be stored and recalled from a dot subdirectory of the home directory of the user. This would provide adequate functionality without adding a keyboard, mouse and display to the system and without relying of special software in the user's computer.

A GUI with feedback for level metering, state of the mixer, etc., would also be very useful (and could potentially make it easier to code a preset management interface).

Once the code stabilizes we think it would also be very useful to add an Openmixer system to our small concert hall, the Stage.

8 Additional Applications

The fact that Openmixer is just a program enables it to be customized for projects that have very special needs. Here is an example:

A study into archaeological acoustics has led Stanford students into researching and questioning the implications of acoustical environments in a 3000-year old underground acoustic temple in Chavín de Huántar, Perú. A group of students explored the effects of reverberant acoustical features of the temple and ran a series of traditional

social scientific experiments to see what the effects of a sound environment are on an individual's ability to complete different types of tasks. Students used a customized version of Openmixer to connect piezo microphones around an apparatus in which subjects were asked to perform tasks on. The signals were convolved (using *jconvolver* as an external program run from within SuperCollider) with synthetic signals emulating certain 3d aspect of the reverberation and presented to subjects in different spatialized settings during the tasks.

9 Conclusions

So far Openmixer has improved the usability of the Listening Room and has opened the door to interesting ways of interacting with space. In particular it now enables very simple multichannel connection through a network jack. Hopefully it will keep making the Listening Room a productive environment for research and music production.

Openmixer is released under the GPL license, and is available from:

https://ccrma.stanford.edu/software/openmixer

10 Acknowledgements

Many thanks for the support of Chris Chafe and Jonathan Berger at CCRMA for this project. Many many thanks also to the Linux Audio user community for the many very high quality programs that have made OpenMixer possible.

References

[1] Stephane Letz, Jack2, http://www.grame.fr/~letz/jackdmp.html

[2] Andre Bartetzki, LAC2007, *"A Software-based Mixing Desk for Acousmatic Sound Difussion"* (http://www.kgw.tu-berlin.de/~lac2007/descriptions.shtml#bartetzki)

[3] Beck, Patrick, Willkie, Malveaux, , SIGGRAPH2009, *"The Immersive Computer-controlled Audio Sound Theater"*

[4] BEAST (Birmingham ElectroAcoustic Sound Theatre), http://www.beast.bham.ac.uk/about/

[5] Fernando Lopez Lezcano, LAC2009, *"The Quest for Noiseless Computers"*

[6] SuperCollider,
http://www.audiosynth.com/
http://supercollider.sourceforge.net/

[7] Fons Adriaensen, AmbDec, An Ambisonics Decoder, http://www.kokkinizita.net/linuxaudio/

[8] Fons Adriaensen, Jconvolver, A Convolution Engine, http://www.kokkinizita.net/linuxaudio/

[9] The Soundwire Project, https://ccrma.stanford.edu/groups/soundwire/

[10] Jacktrip, http://code.google.com/p/jacktrip/

[11] Netjack, http://netjack.sourceforge.net/